INSTALLATION AND SET UP OF R AND ITS AFFILIATED PACKAGES

Marcus Johnson
marcus.johnson@ec.gc.ca

Created: January 11, 2024
Updated: January 15, 2024

BACKGROUND

R is a programming language that can be used for several different applications, including bioinformatics, statistical analysis, and data visualization. R is open source and maintained by the Comprehensive R Archive Network, or CRAN.

By default, R comes with numerous functions that can conduct rudimentary data analysis and basic mathematic functions, but the R programming language is also supplemented by a large number of extension packages for specialized applications that contain reusable code and functions that can be used in your own work. These may be published by CRAN or by third parties, and will come with documentation explaining their use.

Before you conduct any analysis with R, you will need to install R and its companion software.

A summary of these instructions, as well as a quickstart guide, is provided at the end of this document.

CONTENTS

1. INSTALLATION 2
    1.1 Installing the R programming language 2
    1.2 Installing RStudio 4
2. BASIC OPERATIONS OF R 5
    2.1 Storing objects in variables 5
    2.2 Functions 6
    2.3 Explaining your code using comments 7
    2.4 If, else, and for loops 7
    2.5 Working in a folder and importing data 7
3. INSTALLING PACKAGES 8
    3.1 Pre-installed packages 8
    3.2 Extra packages provided by CRAN 9
    3.3 Installing packages from third-party sources 9
4. R FORMATTING AND STYLE GUIDE 9
5. EXTRA RESOURCES 10
6. QUICKSTART 10
BIBLIOGRAPHY 10

1. INSTALLATION

1.1 Installing the R programming language

Installation of most programs on your Environment and Climate Change Canada provided computer is managed by Software Center. Open the Software Center and search for “R for Windows”.

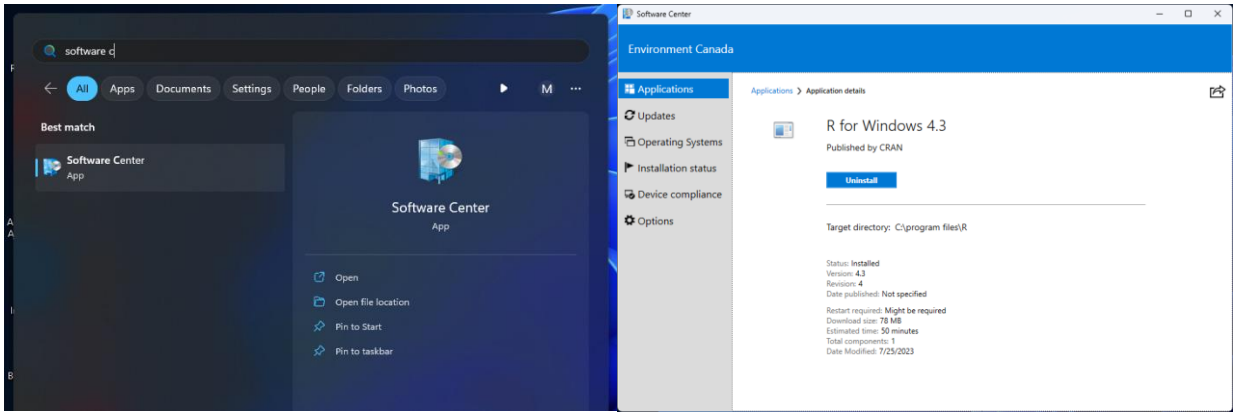

Figure 1: A search for the Software Center in the Windows Search bar, and the R for Windows entry in the Software Center.

R can be installed and run with the default settings. This version of R can be opened by searching in the Windows Search bar for your R version. Opening this basic version of R will open its console, a command line interface that will allow you to enter commands to perform functions.

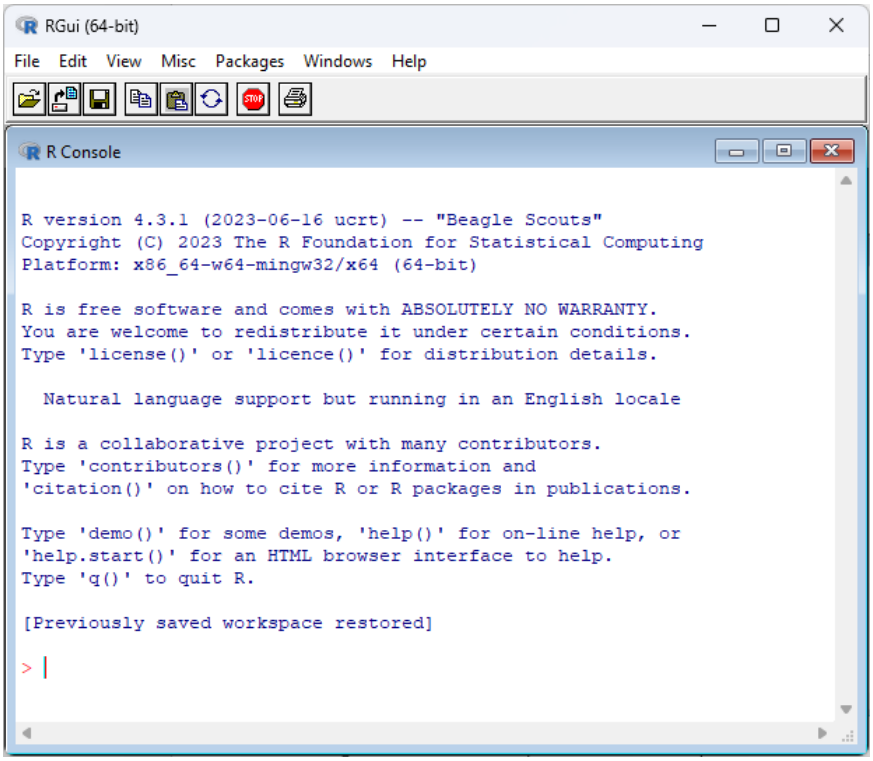

Figure 2: The R terminal.

You can run functions and execute code entered into the terminal from the consol:

```
4*3
[1] 12
sqrt(25)
[1] 5
```

R, by default, also contains example datasets. The iris dataset was originally published in 1936 and describes multiple measurements of petal and sepal of 150 iris flowers.[1]

The `head()` function can be used to see the column names and first 6 lines of the dataset:

```
head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

A basic plot can be generated using this dataset:

```
plot(iris$Sepal.Length, iris$Sepal.Width)
```

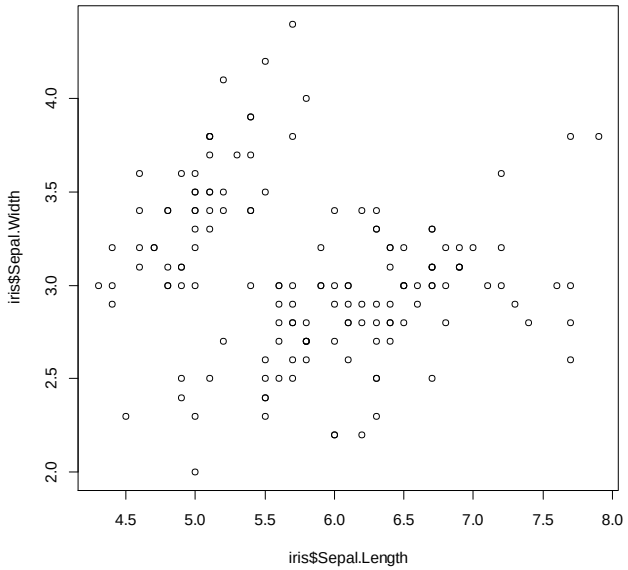

Figure 3: The plot of sepal length by sepal width of the example iris dataset in R.

1.2 Installing RStudio

However, the terminal is a rudimentary way of interacting with the R programming language. In addition to R, we will install RStudio.[a] RStudio is an integrated development environment for R maintained by Posit. It can be installed by the same procedure as installing R, from the Software Center.

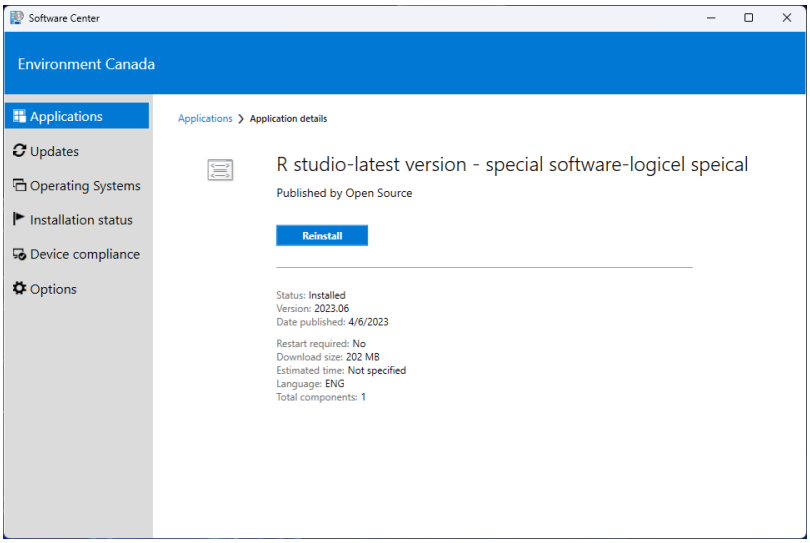

Figure 4: The installation page of RStudio on Software Center, as of January 12, 2024.

RStudio can be opened with the default settings, using the R version previously installed.

Opening the RStudio program shows a more complex environment with console on the left side the screen that operates in the same way as the original R console, but now with a files and plots window in the bottom right and an environment window in the top right. Creating a new R script by `File > New File > R script (Ctrl+Shift+N)` gives access to a fourth window in the top left.

You can consider "R scripts" versions of documents that can be saved, edited, and shared. R scripts have the filename extension ".R" in your file explorer and, if not already done be default, can be set to open in RStudio automatically.

Your version of RStudio may look different but will still operate on the same R code. RStudio appearance and other settings can be found under `Tools > Global Options…` .

Creating and executing code in RStudio can be accomplished by writing code in the R script window and then executing the line by either clicking the **Run** button, or by the keyboard shortcut `Alt+Enter`.

The same plot of sepal length by width can be created by executing the code from earlier. Now, the plot will appear in the bottom right of the screen. The interface there allows the user to right click and save or copy, or export the image directly.

[a] RStudio runs on top of R, a version of R will need to be installed for RStudio to run. This was completed during the initial installation of R.

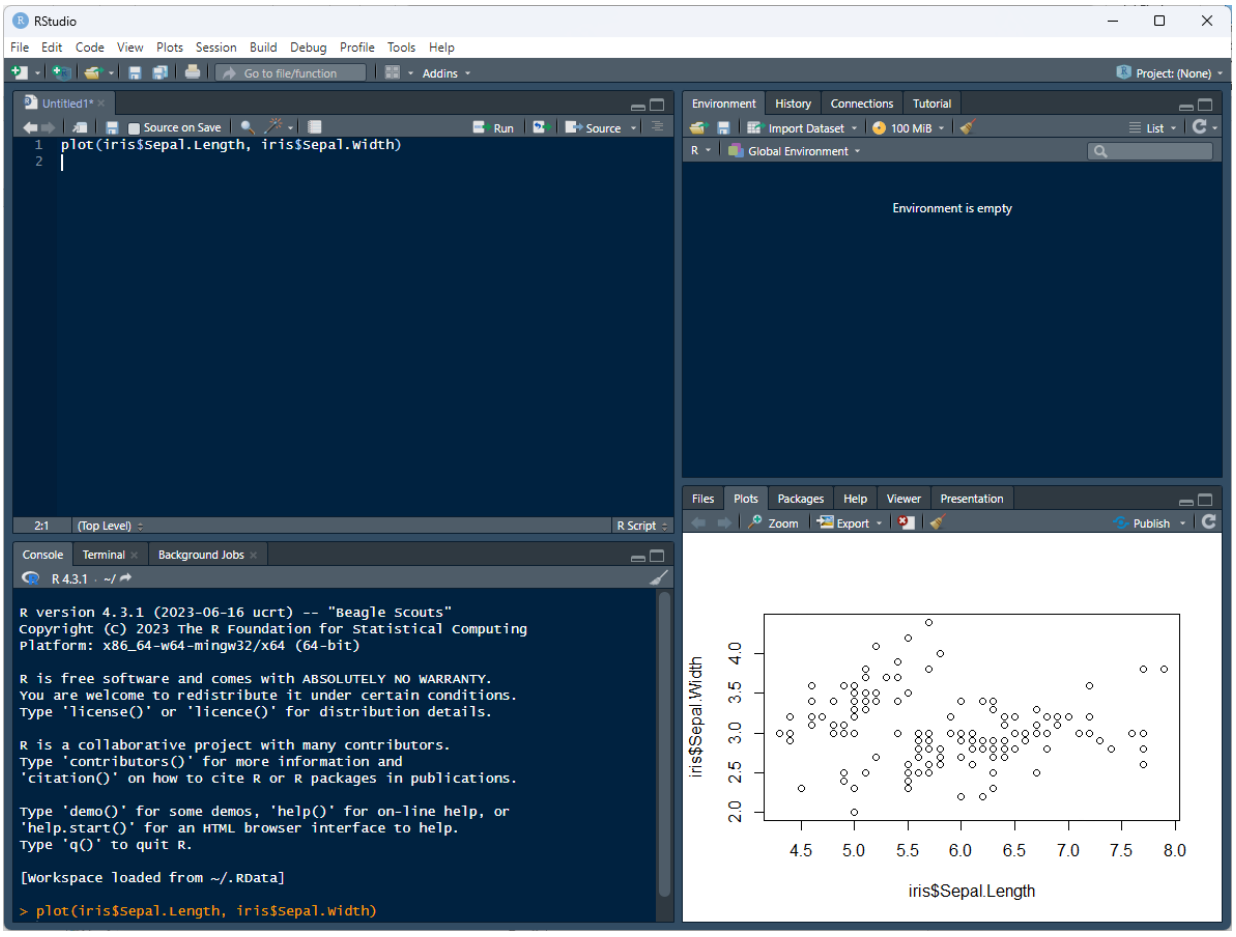

Figure 5: A default RStudio window with (clockwise from top left) the script window, global environment, file browser, and console. Code in the top left in the R script window, executed in the console in the bottom left, with the resulting plot in the bottom right.

2. BASIC OPERATIONS OF R

2.1 Storing objects in variables

R is a programming language that uses objects, variables, and functions. In general, R objects are defined right to left with the <– operator.

R allows you to define variables and call them back later.

```
x <- 5
x
[1] 5
```

Variables can be manipulated in the same way as the objects that define them.

```
y <- 6
z <- 7
y + z
[1] 13
```

These objects can be strings, variables, vectors, data frames (like a table or matrix) or complex objects.

A vector is a list of numbers:

```
list_of_numbers <- c(1, 2, 3, 4)
list_of_numbers
[1] 1 2 3 4
```

Other data types include character strings and logical variables:

```
character_variable <- "Hello, World!"
logical_variable <- TRUE
```

Data frames are commonly used for tabular data like tables or charts:

```
df <- data.frame(Name=c("John", "Jane", "Bob"), Age=c(25, 30, 22))
df
  Name Age
1 John  25
2 Jane  30
3  Bob  22
```

2.2 Functions

R, like other programming languages, also operates by using functions. In principle, a function is an algorithm that accepts arguments or values and returns a result. Earlier, we saw how the function **`head()`** accepted the data iris and performed the function of presenting the top 6 rows. You may have also noticed the **`sqrt()`** function, that accepted a number and returned the square root of the number.

Different functions operate on different objects. The **`sum()`** function is capable of adding all the numbers in a vector or different lists, and **`mean()`** can take the average of a vector.

```
sum(5, 6)
[1] 11
mean(list_of_numbers)
[1] 2.5
```

Be careful, because sometimes functions will operate on different objects in non-intuitive ways.

```
ducks_can_float <- TRUE
witches_can_float <- TRUE
woman_can_float <- FALSE
sum(ducks_can_float, witches_can_float, woman_can_float)
[1] 2
```

2.3 Explaining your code using comments

It is best practice to explain your code while writing your code, within your code. This will assist you or others who review your code in the future.

Comments in R are separated from code using the # number sign. Anything written after # on a line, or a line starting with # will be ignored. This is helpful when you want to make a note of why you are doing something.

```
one_through_six <- c(1:6) # a vector of the numbers one through six
one_through_six
[1] 1 2 3 4 5 6
```

Alternatively, you can write comments ahead of a large section of code.

2.4 If, else, and for loops

R can use `if`, `else`, and `for` loops to construct more complex operations:

```
# Loop through numbers 1 to 6
for (i in 1:6) {

  # Check if the number is divisible by 2
  if (i %% 2 == 0) { # %% is the remainder operator
    print(paste(i, "is divisible by 2"))
  } else {
    print(paste(i, "is not divisible by 2"))
  }
}
[1] "1 is not divisible by 2"
[1] "2 is divisible by 2"
[1] "3 is not divisible by 2"
[1] "4 is divisible by 2"
[1] "5 is not divisible by 2"
[1] "6 is divisible by 2"
```

The more complicated your functions and code becomes, the more important it is to leave clear comments for yourself in the future, or for colleagues who will use or review your code.

2.5 Working in a folder and importing data

The way R operates is that it works in a folder, called your working drive. You can view what drive your working drive is set to when you open an instance of R using the `getwd()` function.

```
getwd()
[1] "C:/Users/JohnsonMa/OneDrive - EC-EC/Documents"
```

Good practice is to save your R script in your working drive for a particular project. Opening RStudio by double clicking on your R script from a folder in the file browser should set your working drive.

Should you need to set your working drive, you can set your working drive to a specific project folder, one that also holds all of your data files. This can be done using the **setwd()** function.

```
setwd("C:/Users/JohnsonMa/OneDrive - EC-EC/Projects/r tutorial")
```

Once you know what folder you are working in, you can open or save items directly to the folder.

```
mtcars_dataframe <- read.csv("mtcars.csv", header = TRUE)
Titanic_df <- read.table("subfolder/Titanic.txt", sep = "\t", header = T)
```

The mtcars and Titanic are other example data frames, like iris, included in the base version of R. Information about those datasets can be accessed using the **?** operator, opening the help window of RStudio.

```
?Titanic
```

The **?** operator will open the help window for any function and should describe its use cases and arguments.

3. INSTALLING PACKAGES

3.1 Pre-installed packages

While R by default comes with a range of functions and operators, extra groups of functions relating to a common theme can be installed and used. These extra functions are grouped in packages and are loaded by calling libraries.

For example, consider a non-normally distributed set of data:

```
x <- c(0.2, 0.528, 0.11, 0.260, 0.091,
           1.314, 1.52, 0.244, 1.981, 0.273,
           0.461, 0.366, 1.407, 0.79, 2.266)
hist(x)
```

To apply the Box-Cox geometric mean transformation[2] to a set of data, the function **boxcox()** cannot be used without first loading the library in which it is stored:

```
boxcox(lm(x~1))
Error in boxcox(x) : could not find function "boxcox"
```

Libraries are loaded using the **library()** function in R, and the **boxcox()** function is stored in the MASS library.[3]

```
library(MASS)
```

Once the MASS library has been loaded, running the **boxcox()** function on the linear model of x by 1 returns the expected transformed distribution of the non-normal data series entered earlier.

```
boxcox(lm(x~1))
```

3.2 Extra packages provided by CRAN

CRAN also hosts extra packages that do not come pre-installed but can be easily downloaded. This can be done with the `install.packages()` function.

Here we will install the tidyverse package,[4] which is actually a collection of libraries all designed to work with "tidy" data.[b]

```
install.packages("tidyverse")
```

Note that when installing packages, the name of the package must be surrounded by quotation marks. Once a package has been installed once, it can simply be called in future to be loaded into the instance of R and does not need to be re-installed.[c]

Once a package has been installed, it can be loaded with the same `library()` function as before.

```
library(tidyverse)
```

3.3 Installing packages from third-party sources

Researchers will also create and share packages, that pertain to different areas of research, who are not affiliated with CRAN and who may host them separate from the CRAN repository. Sometimes these packages require external sources to install their libraries. An example is the DESeq2 package, used to compare high-throughput sequencing assays.[5]

The DESeq2 package can be installed by first installing the "BiocManager" package and using the BiocManager package to install DESeq2.

```
install.packages("BiocManager")
BiocManager::install("DESeq2")
library(DESeq2)
```

This is only an example; packages may be hosted in other places such as GitHub or other repositories. Care should always be taken to ensure that when installing software from third parties, only install from trusted sources.

4. R FORMATTING AND STYLE GUIDE

Programming in any computer language is like using a different language; different languages come with general rules. With the rules of using the code explained above, there are also common styles that help with understanding the language used. This can be compared to using correct punction in another written or spoken language. Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread.

---

[b] "tidy" data describes a data storage format for observations where, simply put, each observation is a single row of information on a table, instead of a matrix of information of different variables by samples.
[c] Sometimes packages will receive updates or patches to work with newer versions of R and may need to be updated or re-installed.

In general, there are lots of resources online to show how to write good R code. The tidyverse style guide at https://style.tidyverse.org/ or the Google style guide at https://google.github.io/styleguide/Rguide.html are two common resources that explain writing good R code.

In general, it is not necessary to follow these style guides to the letter. However, the best practice is to be consistent with your code to make it easy to understand consistently.

5. EXTRA RESOURCES

Posit, the company that maintains RStudio, also hosts a collection of "cheatsheets" for quick reference to common R codes and problems at https://posit.co/resources/cheatsheets/.

6. QUICKSTART

R as a programming language needs to be installed on your Environment and Climate Change Canada computer. This is conducted by the Software Center.

RStudio is an integrated development environment for combining the console, R scripts, plots and help. RStudio can be installed in the same fashion from the Software Center. RStudio works on top of R and requires a base version of R to run.

R uses variables, objects, data frames, and functions. R can use if, else, and for loops to create complex tasks. R works in a folder, that can read and save from within the folder. It is good practice to comment your code to make it easier to understand by you and others in the future.

Libraries provide access to extra functions and can be downloaded from various sources. Help can be accessed using the **?** operator within the console.

BIBLIOGRAPHY

(1) Fisher, R. A. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics* **1936**, *7* (2), 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.
(2) Box, G. E. P.; Cox, D. R. An Analysis of Transformations. *Journal of the Royal Statistical Society. Series B (Methodological)* **1964**, *26* (2), 211–252.
(3) Ripley, B.; Venables, B.; Bates, D. M.; Hornik, K.; Gebhardt, A.; Firth, D. Support Functions and Datasets for Venables and Ripley's MASS, 2023. https://cran.r-project.org/package=MASS.
(4) Wickham, H.; Averick, M.; Bryan, J.; Chang, W.; McGowan, L. D.; François, R.; Grolemund, G.; Hayes, A.; Henry, L.; Hester, J.; Kuhn, M.; Pedersen, T. L.; Miller, E.; Bache, S. M.; Müller, K.; Ooms, J.; Robinson, D.; Seidel, D. P.; Spinu, V.; Takahashi, K.; Vaughan, D.; Wilke, C.; Woo, K.; Yutani, H. Welcome to the Tidyverse. *Journal of Open Source Software* **2019**, *4* (43), 1686. https://doi.org/10.21105/joss.01686.
(5) Love, M. I.; Huber, W.; Anders, S. Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2. *Genome Biology* **2014**, *15* (12), 550. https://doi.org/10.1186/s13059-014-0550-8.